第3章

cosテーブルの線形補間とローパス・フィルタのくふう

Design Wave設計コンテスト2005 Student部門第1位

チーム スーパー最迅（小川陽太，若林秀明）

Design Wave設計コンテスト2005のStudent部門第1位の設計を紹介する．課題で示されたPLL方式をベースにしている．線形補間を用いることで，比較的回路規模が大きいcosテーブルを小さくした．また，ローパス・フィルタのくふうやレシーバ回路のパイプライン化により，FMレシーバ回路全体の規模を抑え，高速化を図った．（編集部）

わたしたちの所属している研究室では，ハードウェアを設計して数値シミュレーションを高速化する研究などを行っています．研究の中には，HDLで回路を設計し，FPGAに実装して動作させるといったこともあります．

昨年のコンテスト（Design Wave設計コンテスト2004 Student部門/LSIデザインコンテスト in 沖縄 2004）では研究室の先輩が準優勝を収めました．そこで，今年もコンテストに応募して沖縄の発表会を目ざしました．また，コンテストを通してハードウェア設計の技術を高められると考え，課題のディジタルFMレシーバの設計にチャレンジしました．

PLL方式のFMレシーバ回路を設計する

まず，設計仕様書に示されていた復調回路をVHDLで記述することにしました．HDLによるハードウェア設計は初めてだったので，とにかく復調できる回路を完成させることを最初の目標にしました．そして，回路を設計していく過程で何かおもしろいアイデアが浮かんだらそれを取り入れていくことにしました．

● PLL回路の設計

設計仕様書に従って最初に設計したFMレシーバのブロック図を**図1**に示します．このFMレシーバは，入力信号に追随して同期するPLL（phase-locked loop）回路を用いて復調を行っています．回路全体は，大きく分けて，位相比較器，ループ・フィルタ，NCO（numerical controlled oscillator：数値制御発振器），LPF（ローパス・フィルタ）の四つのブロックから構成されています．

FMレシーバ回路の完成後，論理合成とシミュレーショ

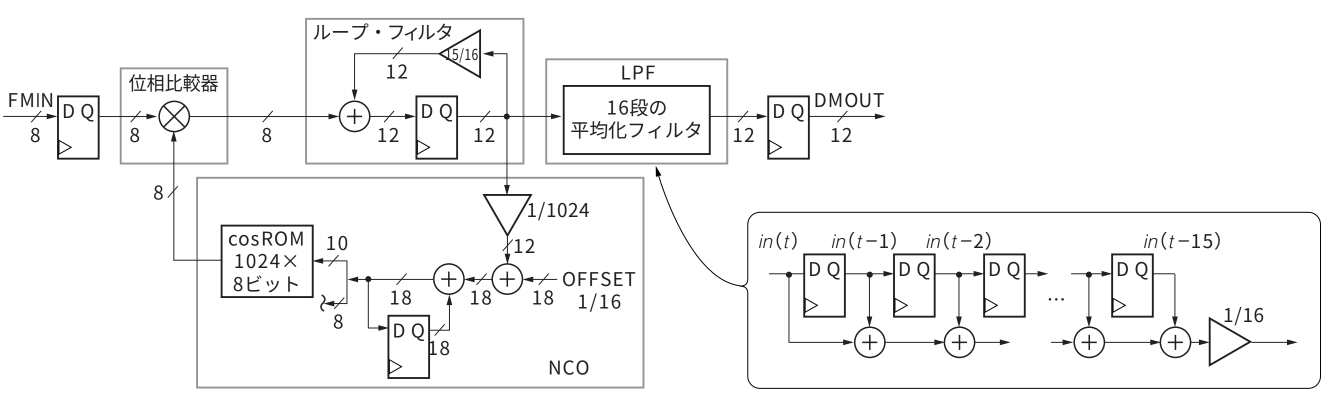


図1　最初に設計したFMレシーバ

設計仕様書に例示されている，PLLを用いたFMレシーバ回路．

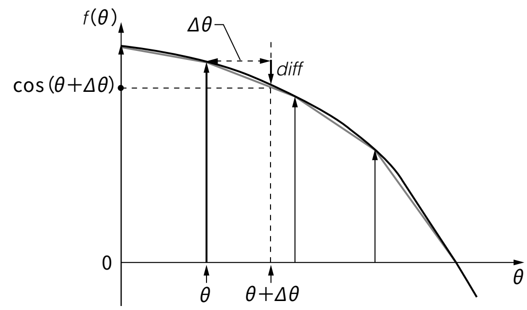
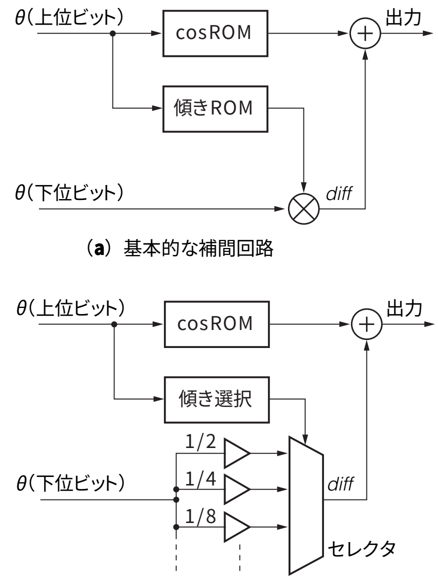


図2　線形補間の方法

cosの曲線をいくつかに分割し，間を直線でつないで近似する．*diff*の値は，直線の傾き×⊿θで計算することができる．



(**a**) 基本的な補間回路

(**b**) 乗算器を使わない補間回路

図3　補間回路のくふう

(a)が最初に考えた補間回路．ここで傾きをビット・シフトで計算できる値に限定し，回路を(b)のように変更することで，乗算器を使わないようにすることを考えた．

ンを行い，課題の矩形波と三角波が正常に復調できることを確認しました．

この回路をベースにして，回路規模の縮小と高速化を図ることにしました．今回は，復調方式と基本的な回路構成は設計仕様書のままとし，各ブロックの内部をくふうすることで回路の効率化を行いました．

● cosテーブルの線形補間

NCOの内部には，cosの値を計算するブロックがあります．設計仕様書に示された回路では，cosの計算に1024×8ビットのROMテーブルを使用していました．そしてこのROMはFMレシーバ全体の中で比較的大きな回路規模を占めていました．

本誌2005年1月号のコンテスト関連記事では，テーブルの補間について紹介されていました(1)．そこで，cosROMに格納する値の個数を減らし，テーブルの間を補間でつなぐことによって，cos計算ブロックの回路規模を縮小させることを考えました．

1）回路方式の検討

補間方式として，アルゴリズムが簡単な線形補間を用いました．これは，**図2**のようにテーブル間を直線とみなして近似する方法で，テイラー展開の1次近似,

$$\cos(\theta+\Delta\theta) \quad \cos\theta+(d\cos\theta/d\theta)\Delta\theta$$

の考えかたに相当します．**図2**におけるcos(θ＋⊿θ)の値は，cos θの値と，直線の傾きと幅⊿θを掛けた値*diff*とを足し合わせて求めることができます．

このアルゴリズムをもとに最初に設計した回路が**図3(a)**の回路です．cos波形の1周期をいくつかに等分して，とびとびのθに対応するcosの値とそのθ付近のグラフの傾きをROMに格納しておきます．次に，入力θの上位の数ビットを使って，cosの大まかな値をROMから取得します．その後，θの下位の数ビットと直線の傾きを掛け合わせて，テイラー展開の1次項に相当する値*diff*を求めます．これを足し合わせてcosの近似値を求めます．

しかし，**図3(a)**の回路では傾きROMから値を取り出した後で乗算と加算を行うため，動作速度が遅くなりそうです．また，cosROMのほかに傾きROMと乗算器が必要になってしまうので，回路規模がそれほど小さくならないのではないかとも思いました．

乗算器や傾きROMを使わずに回路を構成できないかと考えていたところ，ベースとなるPLL回路の設計において各部で乗算器の代わりにビット・シフトを用いたことを思い出しました．そこで，**図3(b)**のように傾きを簡単に計算できる値に限定して，乗算を行う代わりにセレクタでいくつかの傾きから最適な値を選択する，という方法で補間を行うことにしました．

2）精度と誤差の検討

cosROMの容量や回路の詳細を決定するため，精度と誤差について検討しました．

まず，cos波形の1周期を何等分するかについて考えました．cos波形の1周期分の曲線と，それを64等分して間

を直線でつないだグラフを比べてみたところ，見た目はだいたい同じでした．32等分でも全体の見た目はほぼ同じでしたが，傾きが急に変化するところで元波形との違いが大きいと感じました．傾きを限定することも考えて，今回は64等分にしました．

次に，傾きの精度をどこまで落とすことができるかについて考えました．補間アルゴリズムをC言語のプログラムに置き換え，傾きを限定することでどの程度の誤差が出るのかを調べました．その結果，傾きが{ 0，4/16，8/16，10/16，12/16 }の5種類だけでも最大誤差を±1LSB程度にできることがわかりました．この誤差は無視できる程度に小さいと考え，傾きはこの5種類とすることにしました．

その後，傾きの正負について検討しました．cosの波形は，θが0からπまでは単調減少なので傾きはつねにマイナスになります．また，

$$\cos(\theta+\pi)=-\cos\theta$$

の関係があるので，後半の半周期のcosの値は前半の半周期から求めることができます．この関係を取り入れることによって，傾きの符号で場合分けする必要をなくしました．また，必要なROM容量も半分になりました．

3）最終的な回路

最終的には，cos計算ブロックは**図4**のような回路になりました．

まず，入力θの10ビットを上位から1ビット，5ビット，4ビットと分けます．Aの部分で，大まかなcosの値をROMから取り出します．ROMには，cos波形の1周期を64等分したうちの前半の32個の値が格納されています．Bの部分でテイラー展開の1次項に相当する値を求め，ROMから取り出した値と合わせます．最後に，Cの部分で，θの最上位ビットが1の場合は値の符号を反転してから出力します．2の補数形式で表現された値の符号の反転は，全ビットの否定に1を加えることで行えますが，回路規模を削減するためにここで1を加えることを省略しました．

以上のくふうによって，cos計算回路の回路規模を，単純な1024×8ビットのROMを用いた場合と比べて約1/10の大きさにすることができました．実際の値との誤差は＋0.9LSB，－1.9LSB以内です．

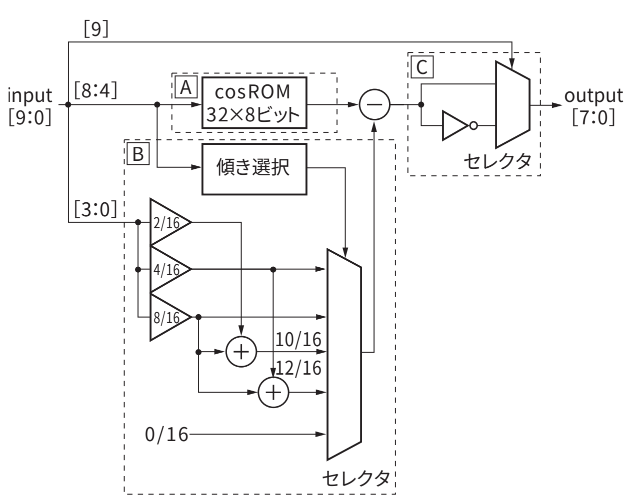


図4　最終的なcos計算ブロック

AのROMには，cos波形の1周期を64等分したうちの前半32個のcosの値が格納されている．Bは補間を行うブロック．Cはcos（θ＋π）を－cos θで求めるブロック．

● ローパス・フィルタの効率化

わたしたちは，**図5**のようなローパス・フィルタを設計しました．この回路は，過去16クロックの間の入力の平均値，

$$output=(in(t)+in(t-1)+\cdots+in(t-15))/16$$

を出力する回路です．フィルタの特性は設計仕様書のローパス・フィルタとまったく同じですが，加算部分をループ状にすることで加算器の数を減らし，回路規模をより小さくすることができました．

回路のAの部分で平均化処理を行っています．まず，初期状態ですべてのDフリップフロップ（DFF）の値を‘0’とします．クロック信号が入るたびに，右側の加算器は入力値を足し合わせて累積していきます．一方，減算器は，16クロック前の入力値を減算します．加算と減算によって，

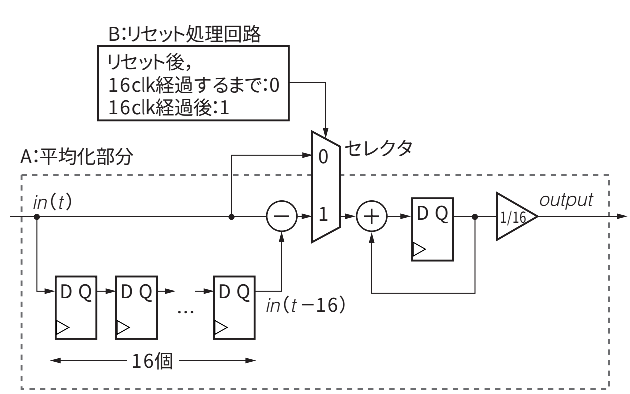


図5　設計したLPF

過去16クロックの間の入力の平均値を出力する回路．加算部分をループ状にすることで，設計仕様書と同じ特性をもつフィルタをより小さな回路面積で実現することができた．

右のDFFの累積値は過去16クロックの間の入力値の合計となります．したがって，この合計を1/16にすれば平均値を求めることができます．

Bの部分はリセット信号を処理するための回路です．図のようにリセット後16クロック経過するまでは‘0’，その後は‘1’を出力する回路とセレクタを使って，リセット後16クロックを経過するまで減算を行わないようにしています．リセット時にすべてのDFFを‘0’にクリアすればBの回路は不要ですが，Bの回路を用いてリセット処理を行ったほうが，結果的により回路規模を小さくすることができました．

このローパス・フィルタ回路を思いつくきっかけになったのは，昨年のコンテスト総評の中で「回路規模を小さくする場合は，回路の共通化を図ります」と説明されていたことです(2)．15個の加算器を共通化できないかと考えていたところ，ループ構造にして加算器を何回も使い回すことを思いつきました．ただ，単にループさせるとどこまでも値が累積されてしまうので，何クロックか遅らせて過去の値を引いていくことを考え，この回路ができました．

● パイプライン化

回路の高速化を図るため，パイプライン化を行いました．

設計したFMレシーバのクリティカル・パスを論理合成ツールが出力したレポートで調べた結果，cos計算ブロックの入力信号とループ・フィルタのDFFの間であることがわかりました．そこでこれらの間にパイプライン・レジスタを挿入することによりパイプライン化を図り，クリティカル・パス遅延を抑えました．

最終的に**図6**のように2ヵ所にレジスタを挿入しています．クリティカル・パスは，IPコアを使用した乗算器となりました．

FPGAに実装して動作させる

今回，設計したディジタルFMレシーバをシミュレーションによって検証するだけでなく，実際にFPGAに実装して動作させることにもチャレンジしました．電子工作を体験してみたいという気持ちもあったので，手配線でボードを製作することにしました．

● FPGAボードの構成

製作したFPGAボードを**写真1**に示します．メイン・ボードには，米国Xilinx社のVertexシリーズのうち「XCV300-4PQ240」と二つの1MビットSRAM（256K×4ビット構成）が搭載されています．このほかにサブボードとして，12ビットD-Aコンバータを搭載したボードと，リセット信号を入力するためのスイッチを搭載したボードを製作し，メイン・ボードに接続しました．

ボードを動作させるためには，電源とクロック信号が必要です．今回用いたユニバーサル基板はパソコンのPCIスロットに挿すことができる形状だったので，パソコンのPCIスロットから5Vの電源と33.3MHzのクロック信号を供給することにしました．

● 自作ボードができるまで

まず最初は，FPGAとLEDだけが載った簡単なボードを作成し必要に応じて後から部品を追加していくことにしました．

FPGAは，研究室で余っていたものを使いました．電圧レギュレータを使ってFPGAで用いる3.3Vと2.5Vの電圧を生成しました．また，FPGAの適当なユーザI/OピンにLEDと抵抗を接続しました．プッシュ・スイッチも搭載し

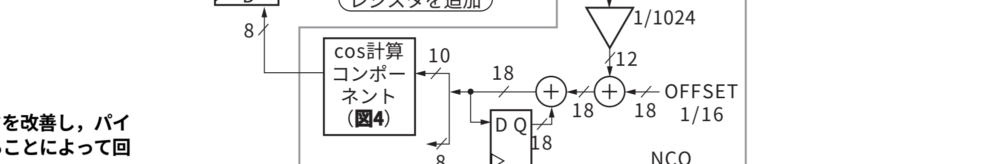
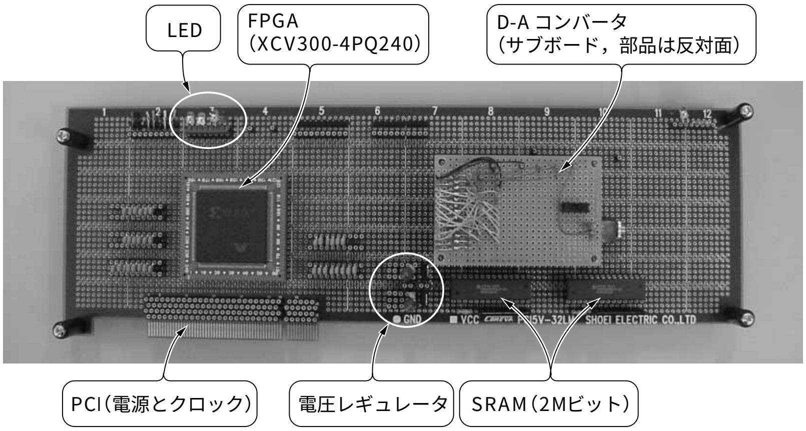
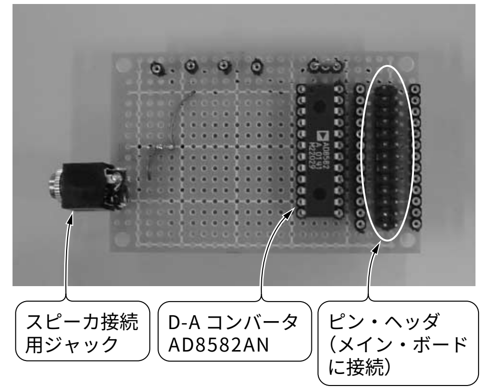


図6
改良後のFMレシーバ
cos計算ブロックとローパス・フィルタを改善し，パイプライン・レジスタを2ヵ所に挿入することによって回路規模と動作速度を改善した．

3



(**a**) メイン・ボード



(**b**) サブボード(D-A変換ボード)

写真1　自作したFPGAボード

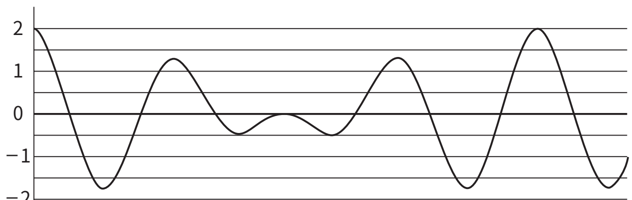


図7　音声を想定した信号の例

cos(440 × 2π*t*) + cos(587 × 2π*t*)の式で表される，音声を想定した信号．

ました．この最低限の回路ができた段階で，LEDの点灯を制御するテスト回路をVHDLで設計してみました．FPGAをコンフィグレーションして，記述したとおりにLEDが光ったときには，とてもうれしく思いました．

FPGAの動作が確認できたので，次にFMレシーバを動作させることを考えました．復調結果を波形として見たいと思ったので，FPGAのディジタル出力をアナログ信号に変換してオシロスコープで確認することにしました．

D-Aコンバータには，米国Analog Devices社の「AD8582AN」を用いました．このD-Aコンバータは動作速度は十分とはいえないのですが，5Vの単一電源で動作し，DIP(dual-inline package)なので扱いやすいLSIです．

さらに，FMラジオ放送のように音声信号を復調して音が出せないかと考えてみました．試しにD-Aコンバータの出力とヘッドホン・スピーカをつないで，FPGAからsin波を出力させてみると，スピーカから音が出ました．そこで，音声を想定したFM変調波を作成してそれを復調させることも試してみることにしました．

音声の変調波をいったんRAMに入れておくとすると，FPGAの内部RAMでは容量が小さすぎます．そこで手元にあった256K×4ビット構成のSRAM(日立製作所のHM624256ALP-20)を二つ載せました．これによって，8ビット幅でアクセスできるようにしました．

2Mビットの容量でも短時間の音声しか復調できませんが，変調波のサンプリング周波数を仕様書の16MHzから260kHzまで落とせば1秒の音声が復調できるので，音を聴いて確認できると思いました．これでFPGAボードは完成としました．

● FM変調プログラムの作成

C言語を用いて，FM変調を行うプログラムを開発しました．これは，任意の信号を周波数変調して2進数8ビットのデータに変換するプログラムです．このプログラムを使って，音声を想定したさまざまな信号のFM変調波を用意しました．例えば，**図7**のように可聴周波数のcos波形を二つ重ね合わせたような信号や，0.6秒のチャイムの音声の変調波を作成しました．

今回は，FPGAボードのRAM容量を考えて，FM変調波のサンプリング周波数を260kHzとしてFM変調波のデータを作成しました．そして，搬送波の周波数を16.25kHz(FM変調波サンプリング周波数の1/16)，最大周波数偏移を162.5Hz(搬送波周波数の1%)としました．この周波数比は設計仕様書の例と同じです．

● FMレシーバ・システムの構成

FMレシーバを自作ボードで動作させるにあたって，FMレシーバとFPGA外部とのインターフェースを制御する

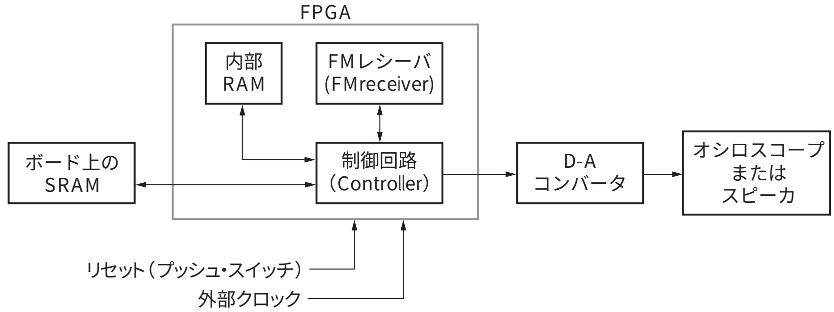


図8
システム構成図
回路モジュールとハードウェアをこのように接続して実際にFPGAボードでFM復調を行い，回路を検証した．

Controllerモジュールを追加して作成しました．そして，**図8**のように回路モジュールとハードウェアを接続しました．

各回路モジュールおよびハードウェアの働きは次のとおりです．

- FMreceiver：今回のコンテストの課題．周波数復調を行う．
- Controller：FMreceiverと外部とのインターフェースを制御する．具体的には，次の三つのことを行う．1）RAMからFM変調データを読み込み，FMreceiverに送る，2）FMreceiverから復調結果を受け取り，D-Aコンバータに送る，3）PCIバスの33.3MHzのクロックを分周し，FMreceiverが用いるクロックを生成する．
- SRAM：FM変調データを格納しておく．
- D-Aコンバータ：12ビットのディジタル信号をアナログ信号に変換する．
- オシロスコープ，スピーカ：復調結果を確認する．

● 動作の流れ

まず，あらかじめ復調したいデータをボード上のSRAMまたはFPGA内部のメモリ(BlockRAM)に格納しておきます．

ボード上のSRAMにデータを格納するときは，いったんFPGAの内部RAMに格納し，それをSRAMに移すという方法をとりました．FPGAのメモリ容量である64Kビットを超えるデータを書き込むときは，データを64Kビットごとに区切って少しずつ格納していきます．データ格納後に別の回路をコンフィグレーションしても，電源を落とさない限り，値はSRAMに保持されます．

RAMにすべてのデータを格納したら，FPGAにFMレシーバ回路をコンフィグレーションした後，プッシュ・スイッチでリセット信号を入力します．これで復調がスタートします．

表1　回路の効率化前と効率化後の比較

● 回路規模(単位：Slice)

	効率化前	効率化後	効率化率
cos計算ブロック	293	26	約1/11
ローパス・フィルタ	236	31	約1/8
回路全体	632	147	約1/4

● 最高動作周波数(単位：MHz)

	効率化前	効率化後	効率化率
回路全体	24.308	51.224	約2倍

Controllerは，RAMのデータを読み込んでFMreceiverに送ります．FMreceiverによって復調が行われ，その結果がControllerに返された後，D-Aコンバータに送られます．時間の経過とともに，FPGAからは次々と結果が出力されます．それをアナログ信号に変換した後，オシロスコープによる波形観測とスピーカからの音声出力の両方で結果を確認しました．

● 論理合成結果

設計仕様書に従って設計した**図1**の回路と，効率化した後の**図6**の回路について論理合成を行い，合成結果レポートにより回路を評価しました．それぞれの回路の回路規模と動作速度は**表1**のようになりました．論理合成ツールにはXilinx社の「ISE 6.2」を使用し，ターゲット・デバイスを今回実装したXCV300-4PQ240に設定して論理合成を行いました．Slice(スライス)とは，Xilinx社のFPGAで回路規模を示す単位です．

比較的大きな回路規模を占めていたcos計算ブロックとローパス・フィルタの回路規模を大幅に減少させることができました．また，VHDLソースの細かな改善などにより，回路のそのほかの部分でも回路規模が少し小さくなりました．回路全体では，約4分の1の回路規模にすることができました．

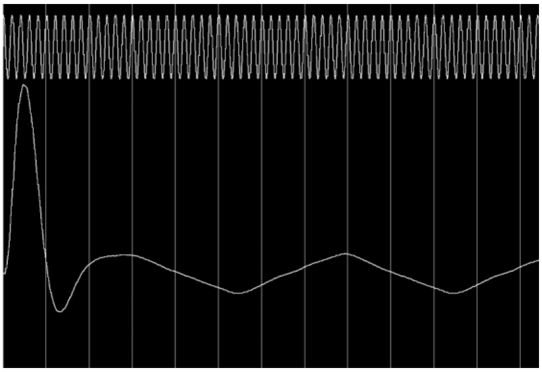
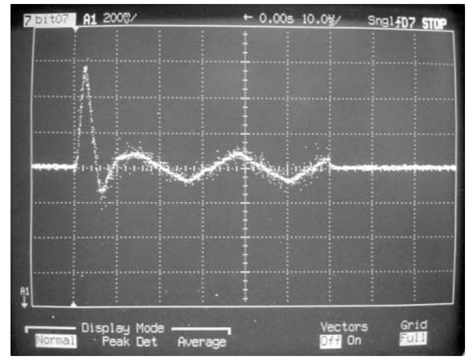
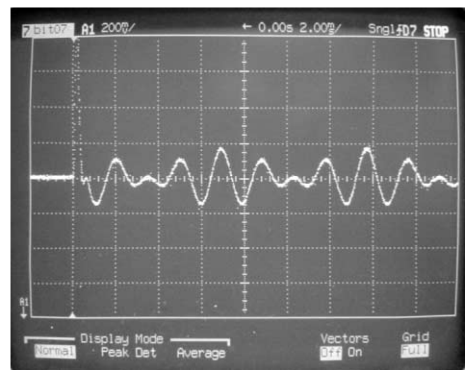


図9　シミュレーションによる復調波形

米国Mentor Graphics社の「ModelSim SE5」を用いて復調のシミュレーションを行った．上段の波形がFM変調データ，下段の波形が復調波．



（a）三角波



（b）図7の信号

写真2　オシロスコープで確認した復調結果

(a)は，課題の三角波を動作速度16.7MHzで復調したときのオシロスコープの波形．1目盛りは横軸が10μs，縦軸が200mV．(b)は，図7の信号を動作速度260kHzで復調したときのオシロスコープの波形．1目盛りは横軸が2ms，縦軸が200mV．

さらに，動作速度を約2倍にすることができました．クリティカル・パスは符号付き8ビットの値どうしを乗算する部分となっています．

シミュレーションおよび実機で動作させたときの回路の出力は，**図9**と**写真2**のようになりました．シミュレーション初期で復調波がオーバシュートしていますが，その後正常に復調できていることが確認できました．

音声データを復調してスピーカから音を発生させることにも成功しました．**図7**に示した二つのcos波を重ね合わせた信号を復調すると，スピーカから二つの高さの音が重なって聞こえました．また，0.6秒のチャイムの音声の変調波を作成してFMレシーバで復調すると，スピーカから元の音声と同じようなチャイムを聴くことができました．

まとめ

任意の音声を自作のFPGAボードを使用して音声として復元するところまで到達できたことに達成感を感じ，回路設計の楽しさを味わうことができました．また，多くの知識を得ることができました．わたしたちにとって今回の課題は，回路設計のおもしろさを知り，回路設計技術や自分自身の向上のさらなる欲求を駆り立てるものであったと思います．

わたしたちはVHDLによる設計もFPGAボードの製作も初めてでした．しかし，しだいにハードウェアやVHDLについての理解が深まり，最終的に音声を復調することまで達成できたことはたいへんうれしく思います．今回のFMレシーバ設計を通して得た知識を，今後のハードウェア設計に生かしていきたいと思います．

今回のディジタルFMレシーバの作成という課題は，非常に奥が深いと思います．実際，今回作成した回路でまだおもしろいことができるのではないかと思っています．

参考・引用*文献

(1) 西村芳一；ディジタルFMレシーバの実現方法を検討する，pp.109-110，Design Wave Magazine，2005年1月号．
(2) 森屋和喜；AES用SubBytes変換回路設計のポイント，p.94，Design Wave Magazine，2004年6月号．

おがわ・ようた
千葉大学工学部 電子機械工学科4年（現在はソニーLSIデザイン）
わかばやし・ひであき
千葉大学工学部 電子機械工学科卒業，研究生（現在は大学院 自然科学研究科 修士1年）